Design and Implementation of a new Inter-Process Communication Architecture for Log-based HIDS for 100 GbE Environments

---

Bachelor Thesis

by

Paul Raatschen

University of Potsdam
Institute for Computer Science
Operating Systems and Distributed Systems

Supervisors:
Prof. Dr. Bettina Schnor
M.Sc. Max Schrötter

Potsdam, April 9, 2023

Raatschen, Paul
raatschen@uni-potsdam.de
Design and Implementation of a new Inter-Process Communication Architecture for Log-based HIDS for 100 GbE Environments
Bachelor Thesis, Institute for Computer Science
University of Potsdam, April 2023

Thanks

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendeten Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Potsdam, April 9, 2023

Paul Raatschen

Abstract

Deutsche Zusammenfassung

Deutsche Zusammenfassung

Contents

1 Introduction

Since the advent of the Internet, bandwiths available to both commercial and private users have been ever increasing. While this opens up new possibilitys for high bandwiths network applications, it also poses new security challenges for dealing with potentially malicious network traffic. In addition to traditional firewalls, Host-based Intrusion Detection Systems (Host-based Intrusion Detection System (HIDS)) are a commonly used security measure, to protect a system . Traditionally, HIDS make use of application logfiles, which are parsed for information on possible attacks and to identify malicious clients. The Introduction Prevention Systems (Intrusion Prevention System (IPS)) Fail2ban[1] is is an one of the most prim

The goal of this thesis will be the design and implementation of a new Inter-Process Communication (IPC) architecture for the transmission of log messages, that is able to facilitate low latency communication between sender and receiver. Additionally, the design should be able to scale to multiple recipients, in order to accommodate more complex security system, in which several processes require access to a hosts application log. For this purpose, a Proof-of-Concept IPS will be developed, that utilizes the proposed IPC architecture to receive log messages and ban malicious clients in the style of Fail2ban.

This thesis will be structured as follows: The following section provides background information on relevant concepts,

2 Background

The following section introduces the concept of HIDS with the specific example of Fail2ban and presents the problem setting this thesis aims to solve. In addition to that, an overview over common types of Inter-Process Communication and existing IPC based logging solution is given. Finally, external libraries and other software used for the implementation and evaluation of the Proof-of-Concept IPS are introduced.

2.1 Host-based Intrusion Detection / Prevention

The idea of specialized software for detecting intrusion attempts and other security threads goes as far back as 1980, when James Anderson published a study on "Computer security threat monitoring and surveillance"[2]. In 1987, Dorothy Denning presented a seminal model for Intrusion Detection Systems, that suggested the use of pattern matching based on statistical analysis of audit records generated by a system, in order to detect abnormal user behavior [3]. Intrusion Detection Systems in general, gather data from a multitude of sources, which is then processed to identify and report potential threads. Host-based Introduction Detection Systems in particular, use information that is provided by the hosts under their supervision. This includes event logs of applications, as well as operating system (Operating System (OS)) based information, such as user logins, file system operations or systemcalls. The analysis of the gathered data can be divided into two categories: 1. Misuse based detection relies on predefined patterns of misuse or malicious behavior, which are then matched against the observed behavior in the data. 2. Anomaly based detection uses statistical analysis to identify deviations from the norm, thereby also being able identify attacks, that have not been previously observed [4]. Intrusion Prevention Systems (IPS) constitute a special class of IDS, which are not only capable of detecting an attack, but also take measure to prevent or mitigate it.

2.1.1 Fail2ban

Fail2ban is a open source IPS for POSIX Systems, that is widely used to protect web servers, for instance against brute-force login attempts, as well as other types of attacks []. To identify potentially malicious clients, Fail2ban makes use of application logs, that are parsed based on a predefined filter. Fail2ban uses configuration units called 'Jails' , that allow for the customization to a wide range of applications. A Jail defines the path to the application log, the filter being applied to the log messages within the logfile and an action, that is executed on client matching the filter criteria. In addition to that, Jails contain further parameters, such as the threshold of matches a client need to reach, in order for the action to be executed, as well as the duration of the action. The filter component of a Jail defines a set of regular expressions, that are used to identify certain events in a log, like an unsuccessful login attempts or the exceeding of a rate

limit. the filter also obtains a clients IP address as well as the date and time of the log messages, to determine, if the event occurred in a relevant time frame.

2.2 Inter-Process Communication

2.2.1 Types of IPC

2.2.2 IPC based logging

Syslog, Rsyslog

2.3 Special Software

2.3.1 Hyperscan

Hyperscan is a open source regular expressions matching engine developed by Intel. It is specifically designed for high performance use cases, such as the application in security contexts and is being used by the intrusion detection systems Snort and Suricata. The process of regular expressions matching with Hyperscan is separated into compile- and run-time. At compile-time a set regular expressions in string representation are compiled into a database, with additional configuration options

2.3.2 io_uring

2.3.3 Trex

3 Design & Implementation

3.1 Requirements

3.2 Abstract Architecture

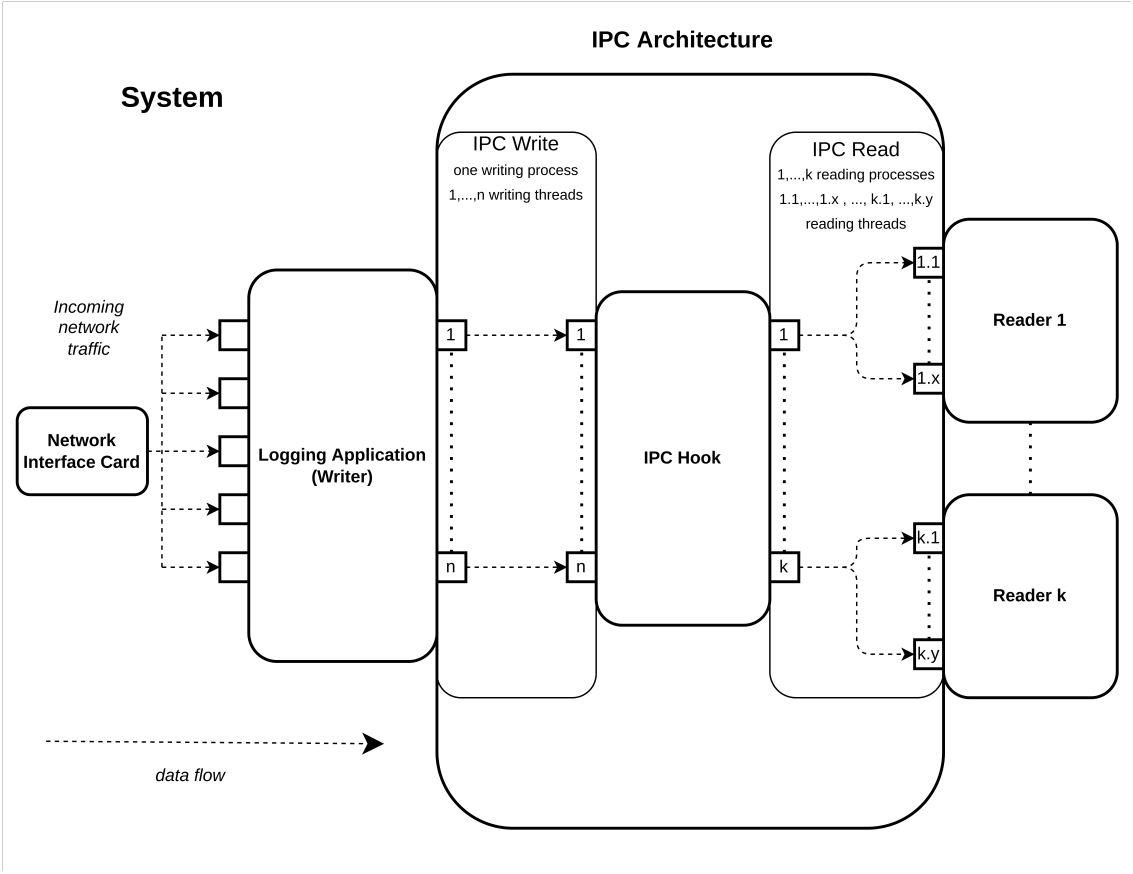
3.3 Choice of IPC Type

3.4 Shared Memory API

3.5 Proof-of-Concept IPS

3.6 Test Application

Figure 3.1: Abstract IPC architecture



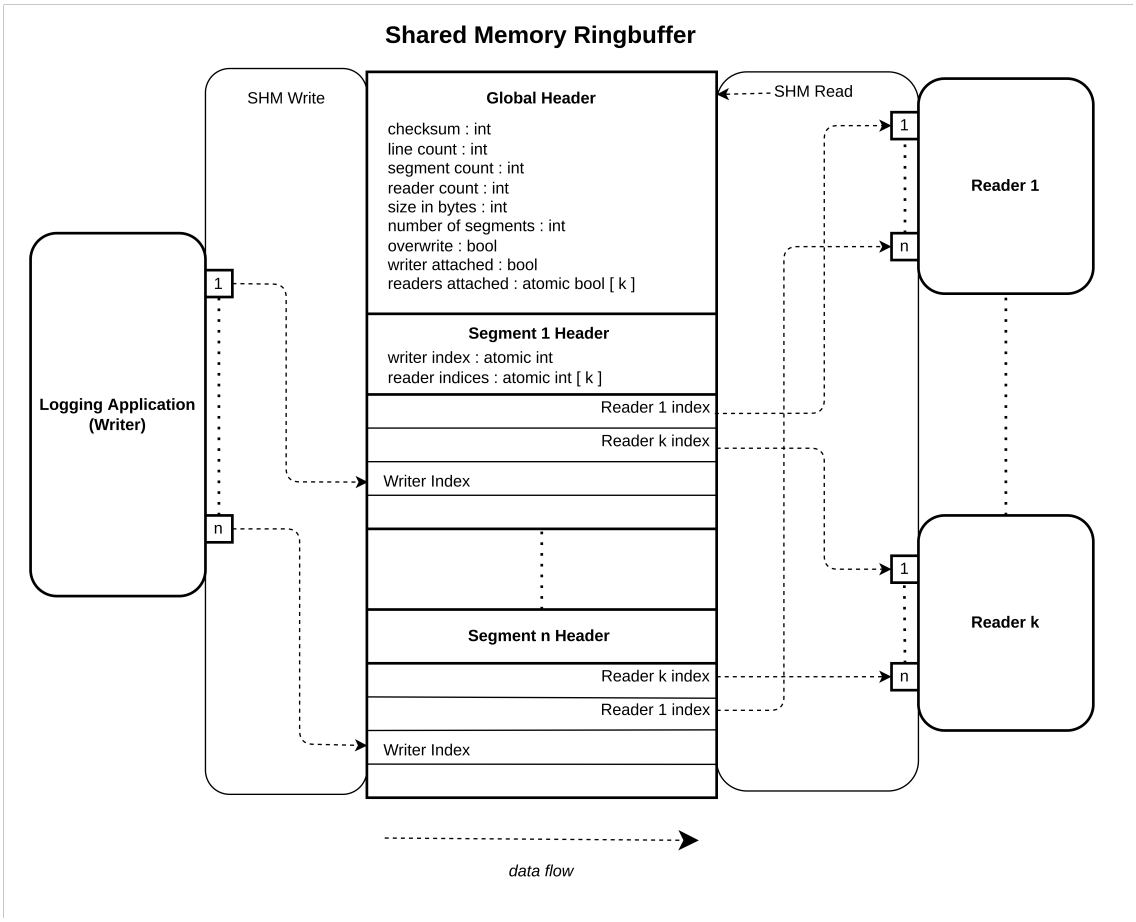


Figure 3.2: Architecture for the single-writer multi-reader shared memory ringbuffer for the transmission of log messages.

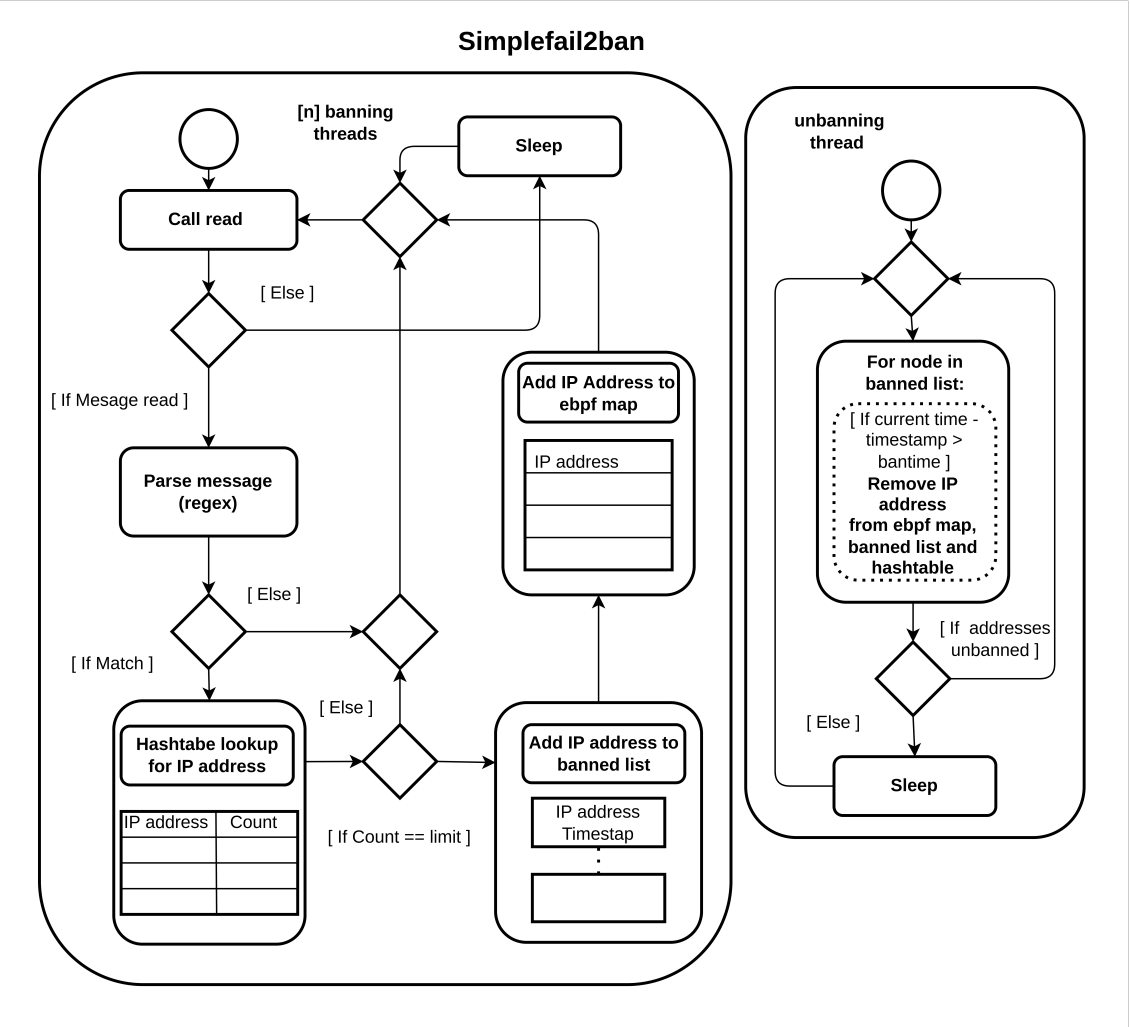


Figure 3.3: Activity diagram for the proof-of-concept IPS implementation. A variable number of "banning threads" receive log messages from a host and parse them with a predefined regular expressions. For messages that match the expression, the clients IP address is extracted from the log message and added to a hashtable, that keeps count of the number of matches per address. If the count reaches the configured limit, the address is added to the list of banned addresses with a current timestamp and inserted into the eBPF map. One "unbanning thread" routinely iterates through the banned list and checks, if a clients bantime hast elapsed. Clients with an elapsed bantime are removed from the eBPF map, banned list and hashtable.

4 Evaluation

4.1 Testenviroment

4.2 Experimental Design

4.3 Fail2ban Replication Measurements

4.4 Simplefail2ban, Logfile Measurements

4.5 Simplefail2ban, Shared Memory Measurements

4.6 Shared Memory Special Measurements

Table 4.1

Hardware	
CPU	Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz
NIC	Mellanox ConnectX-6 100GbE
RAM	128GB
Software	
OS	Debian 11
Kernel	6.1.0-0
NIC Driver	mlx5_core, 6.1.0-0
Fail2Ban	0.11.2
TRex	3.02

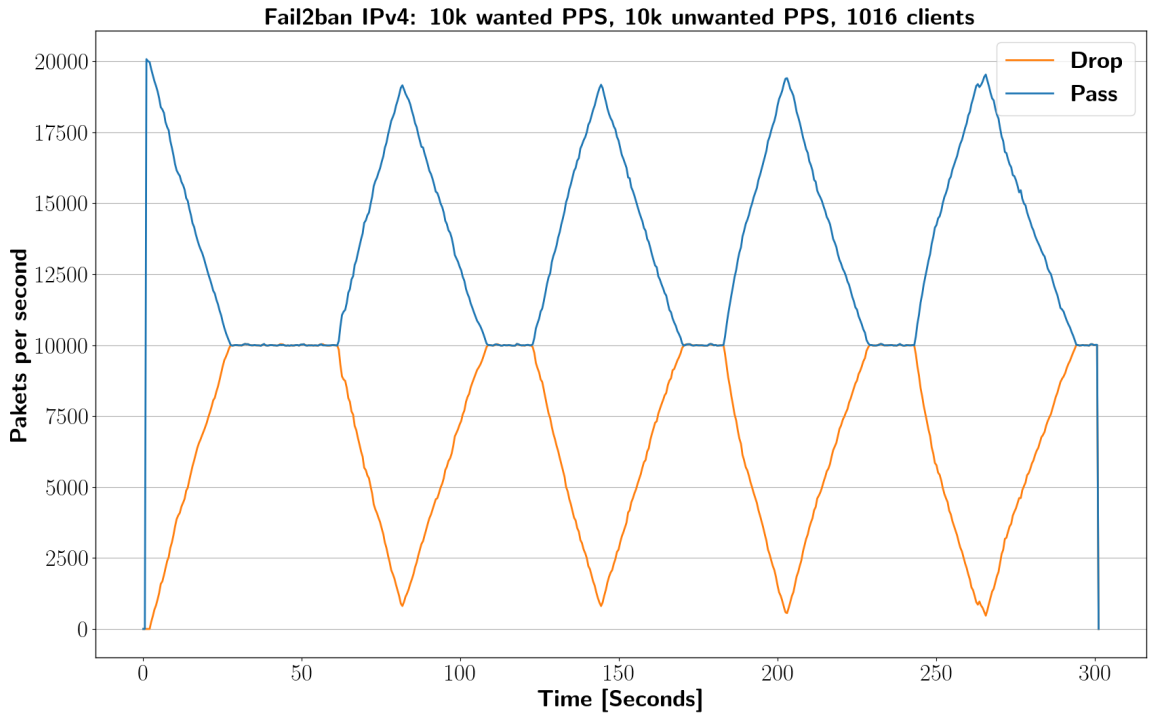


Figure 4.1: Abstract architecture

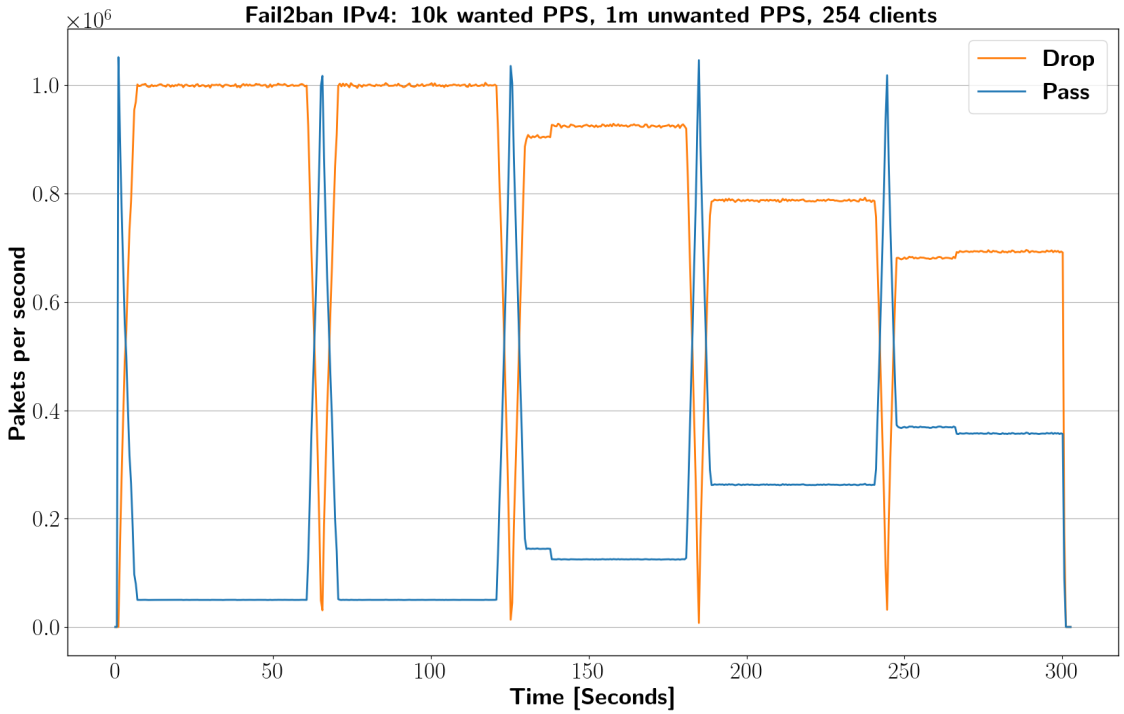
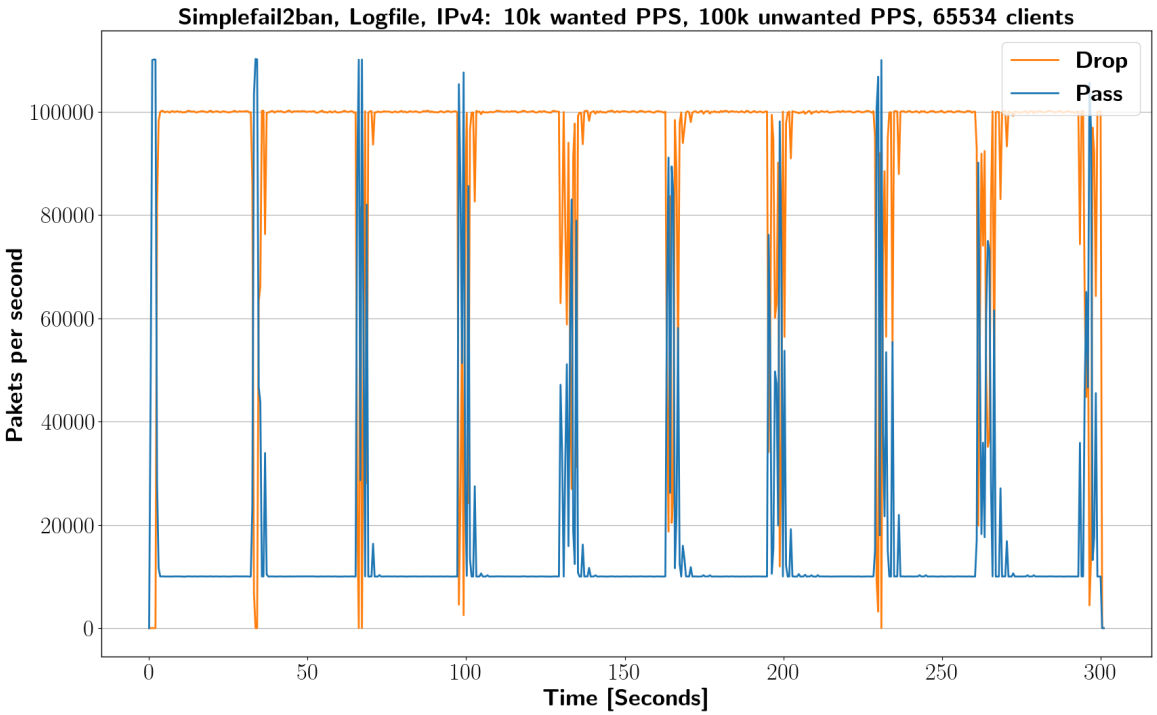
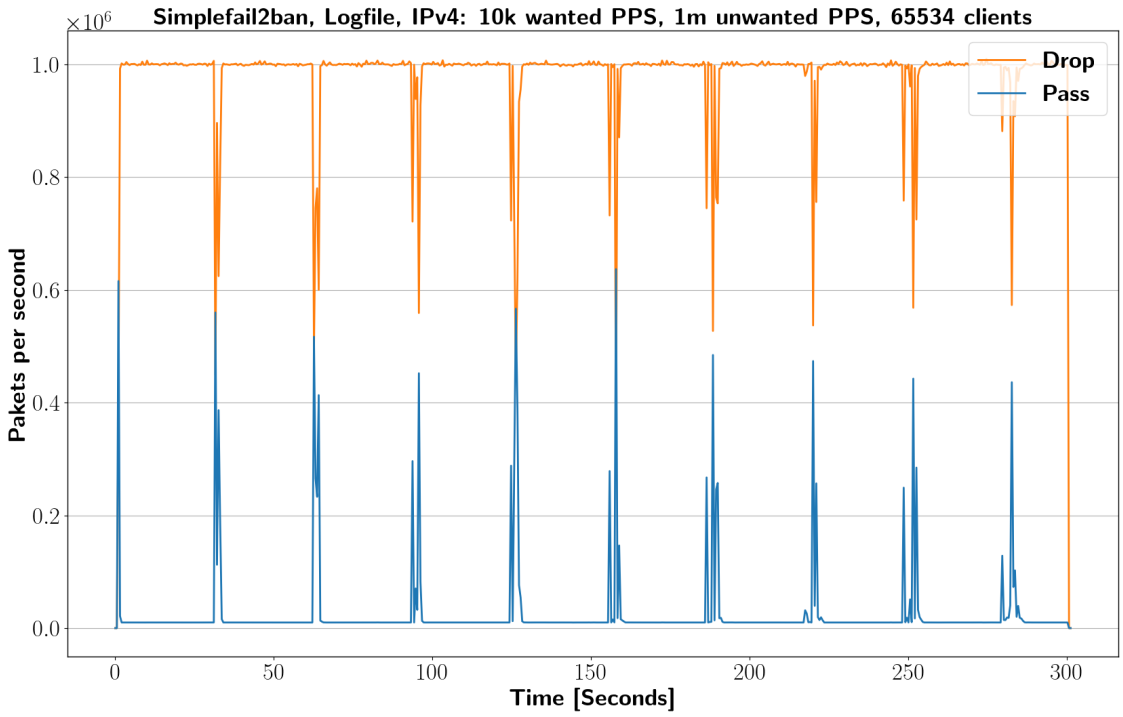


Figure 4.2: Abstract architecture



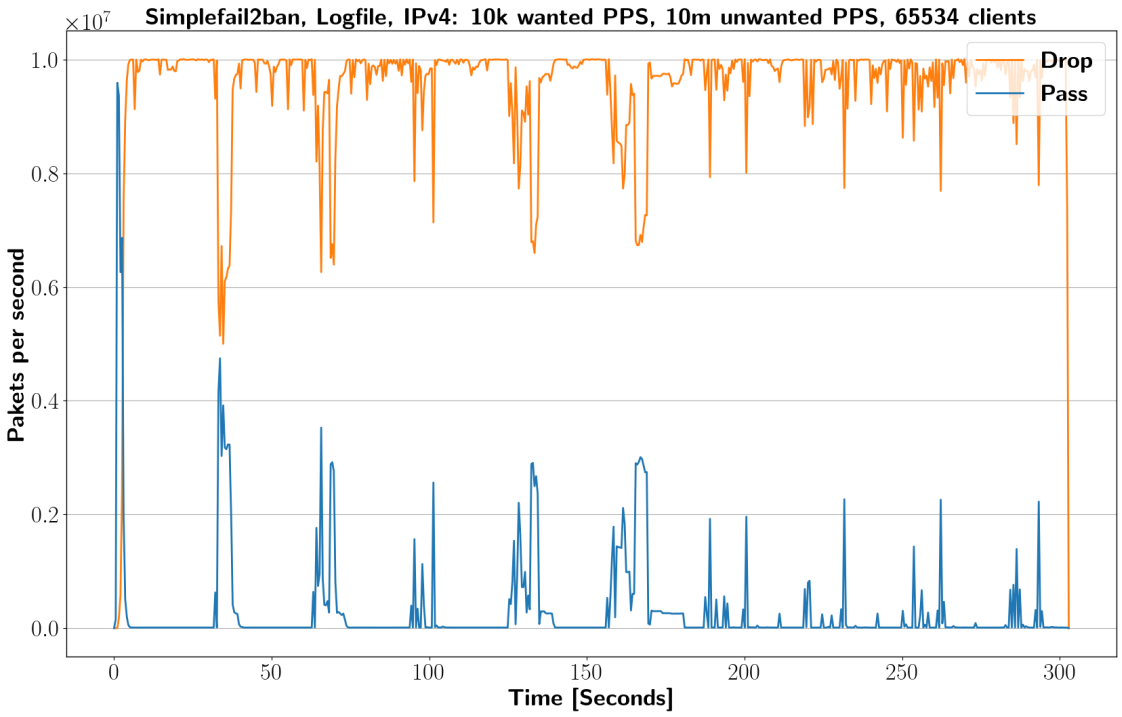
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
33	27.93	99.59	2.07	10.51

Figure 4.3: Some text



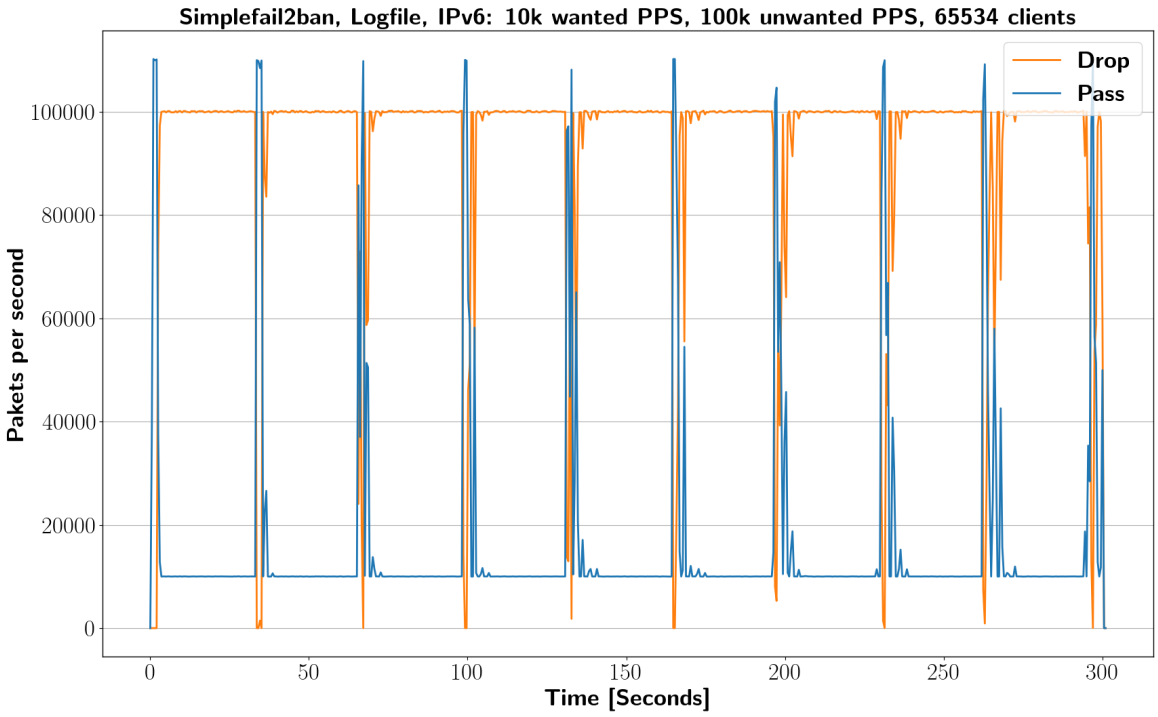
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
303	294.43	98.79	4.7	14.99

Figure 4.4: Some text



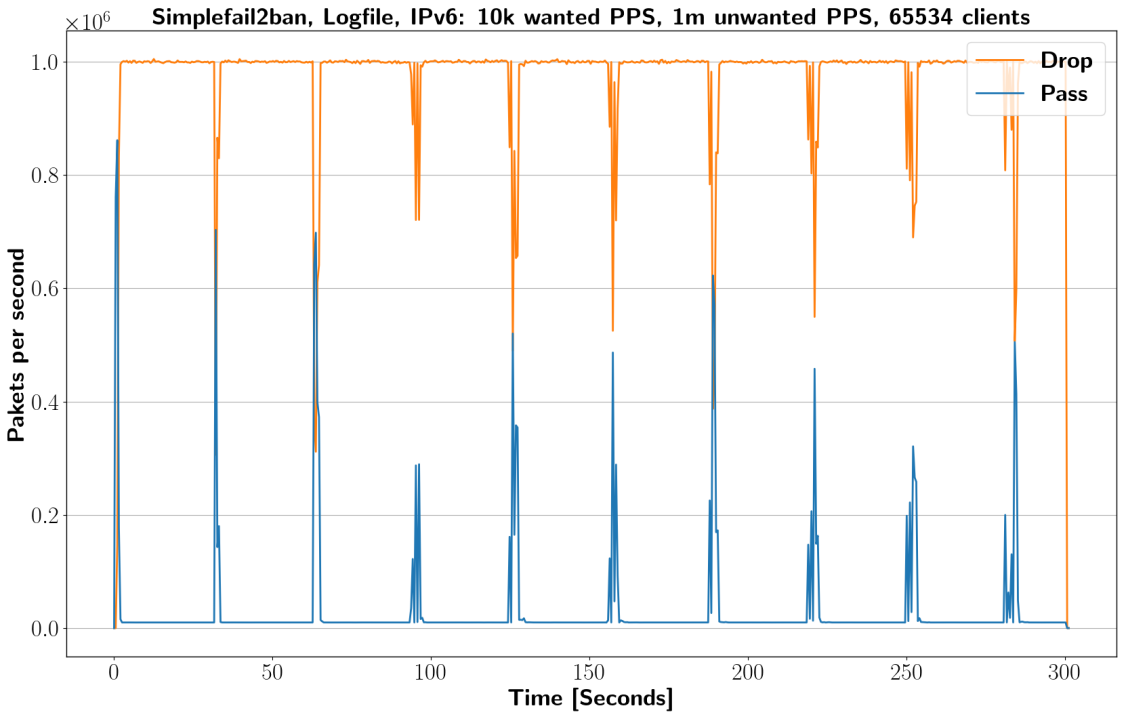
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2986.89	2884.14	96.72	34.39	83.72

Figure 4.5: Some text



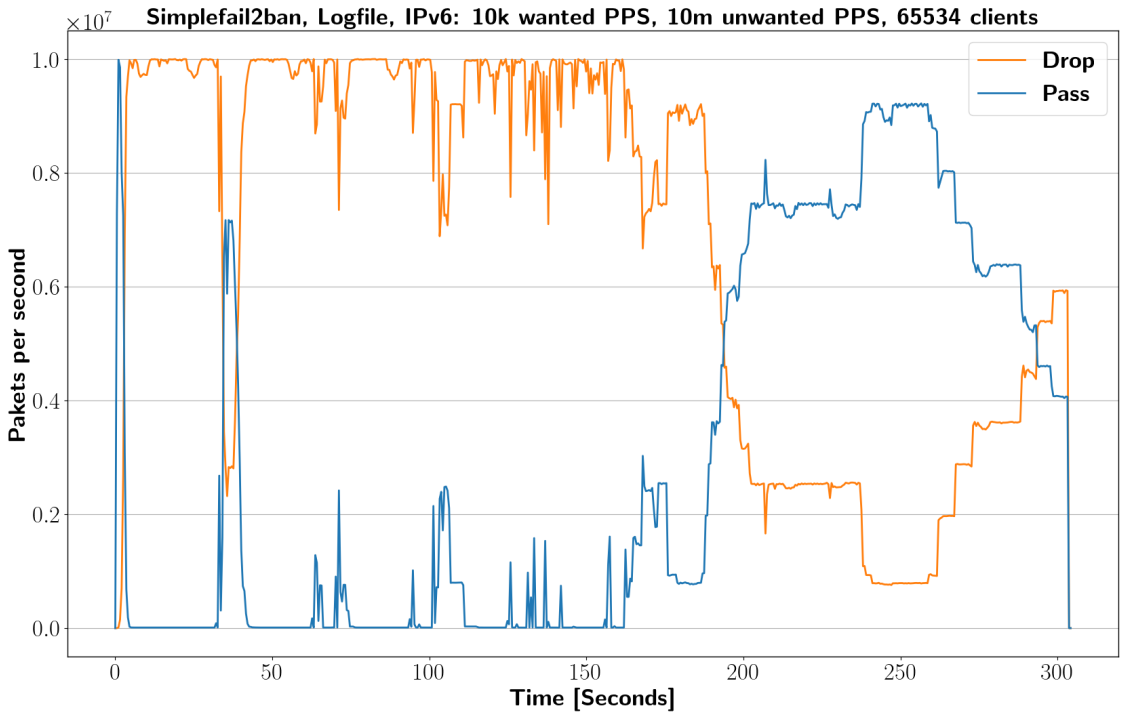
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
33	27.96	99.67	2.04	11.75

Figure 4.6: Some text



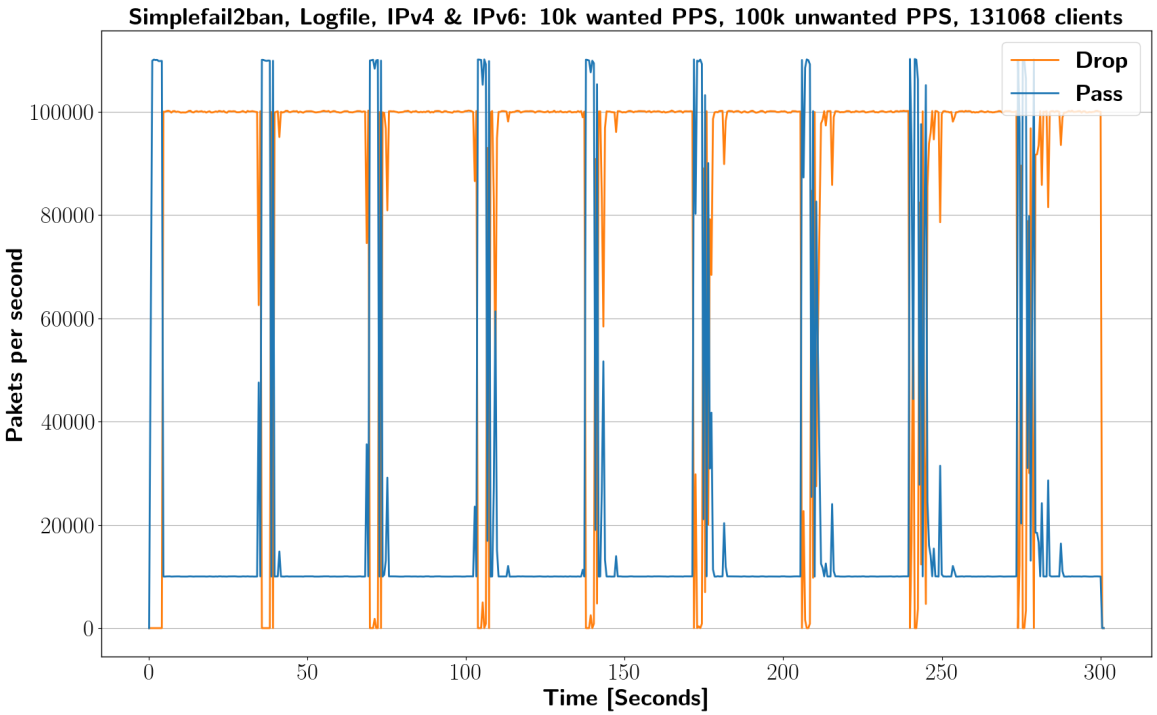
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
303	293.49	98.48	5.58	21.08

Figure 4.7: Some text



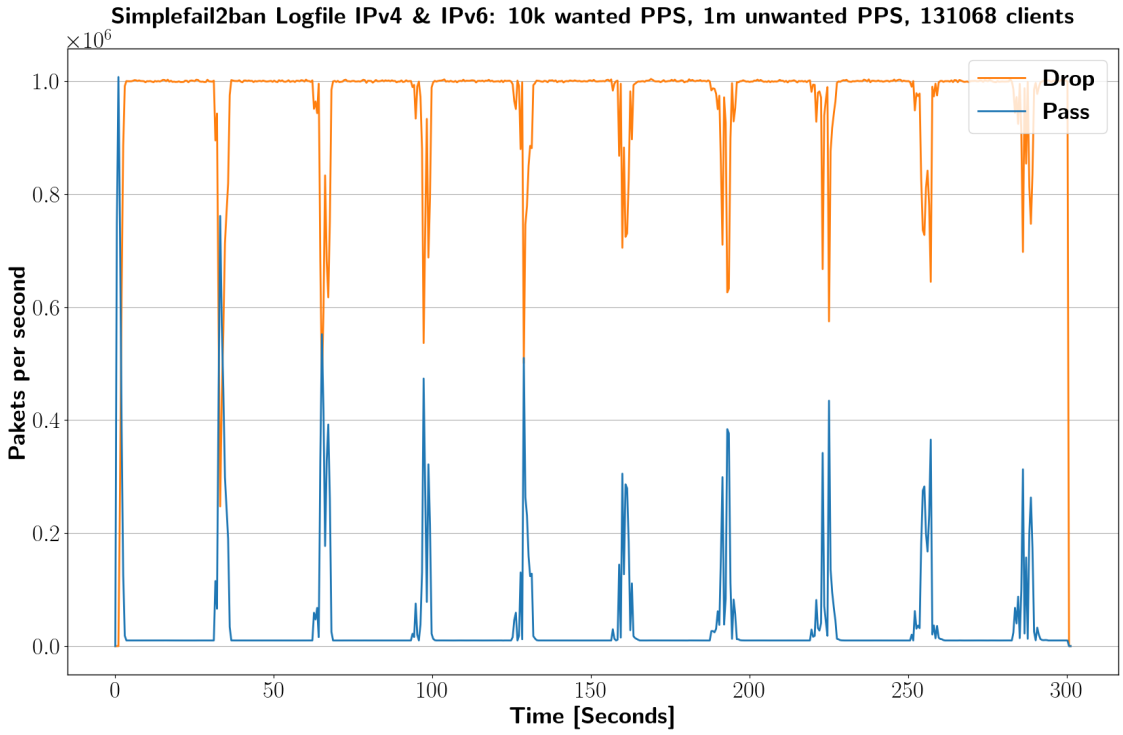
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2992.43	2065.28	69.13	205.4	346.73

Figure 4.8: Some text



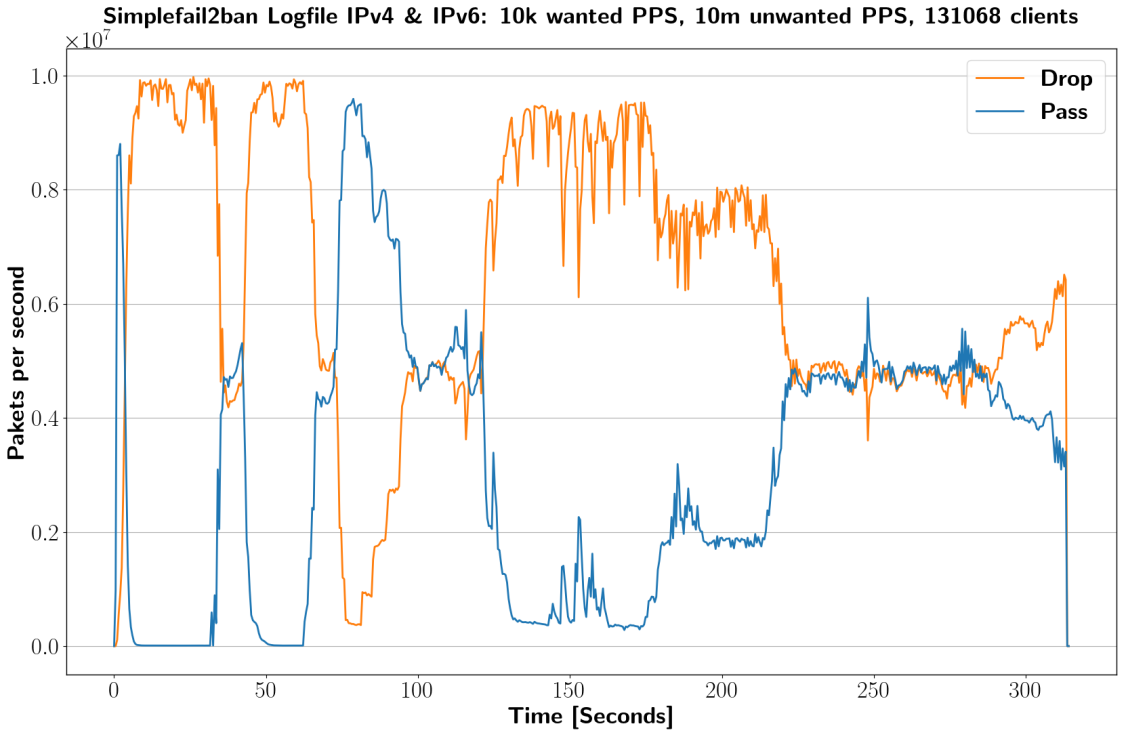
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
33	26.45	99.96	3.55	18.47

Figure 4.9: Some text



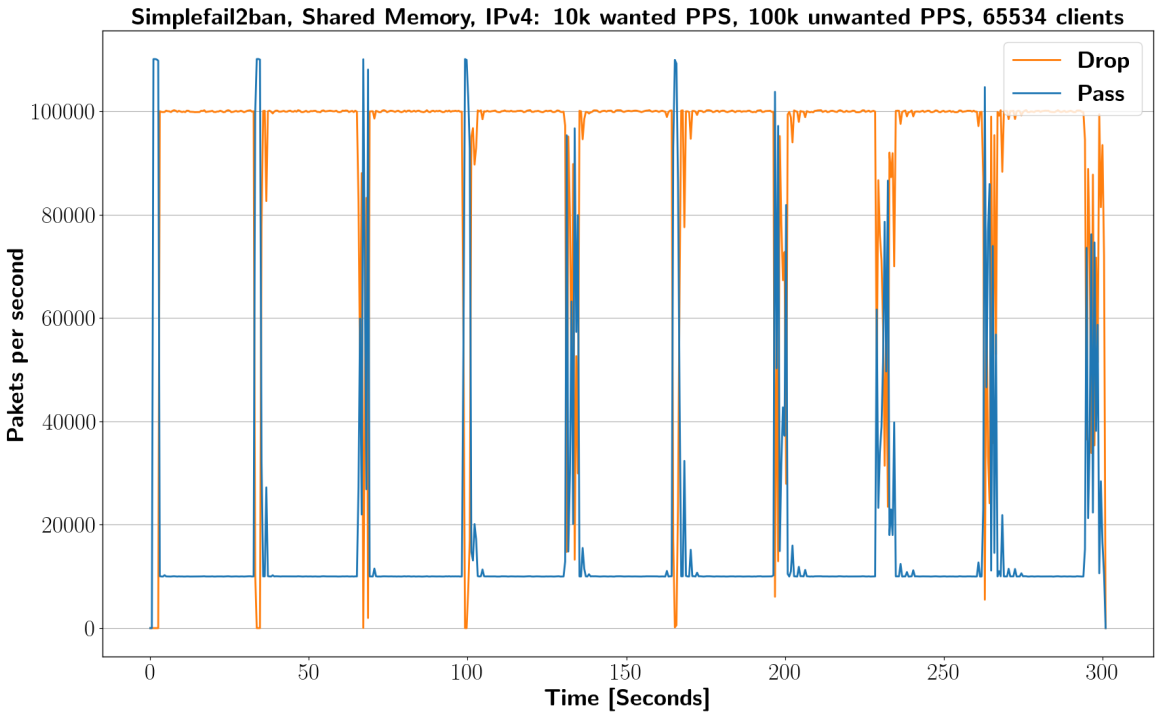
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
303	290.47	98.11	8.5	32.94

Figure 4.10: Some text



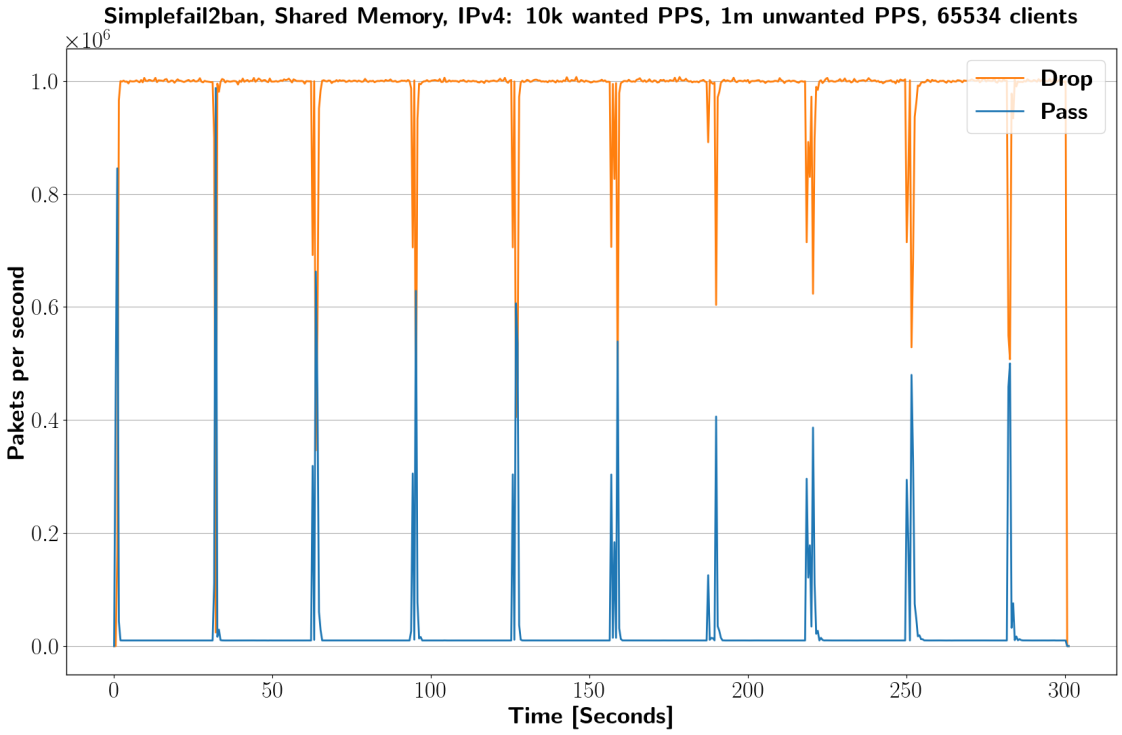
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2993.57	2014.77	67.46	213.63	382.19

Figure 4.11: Some text



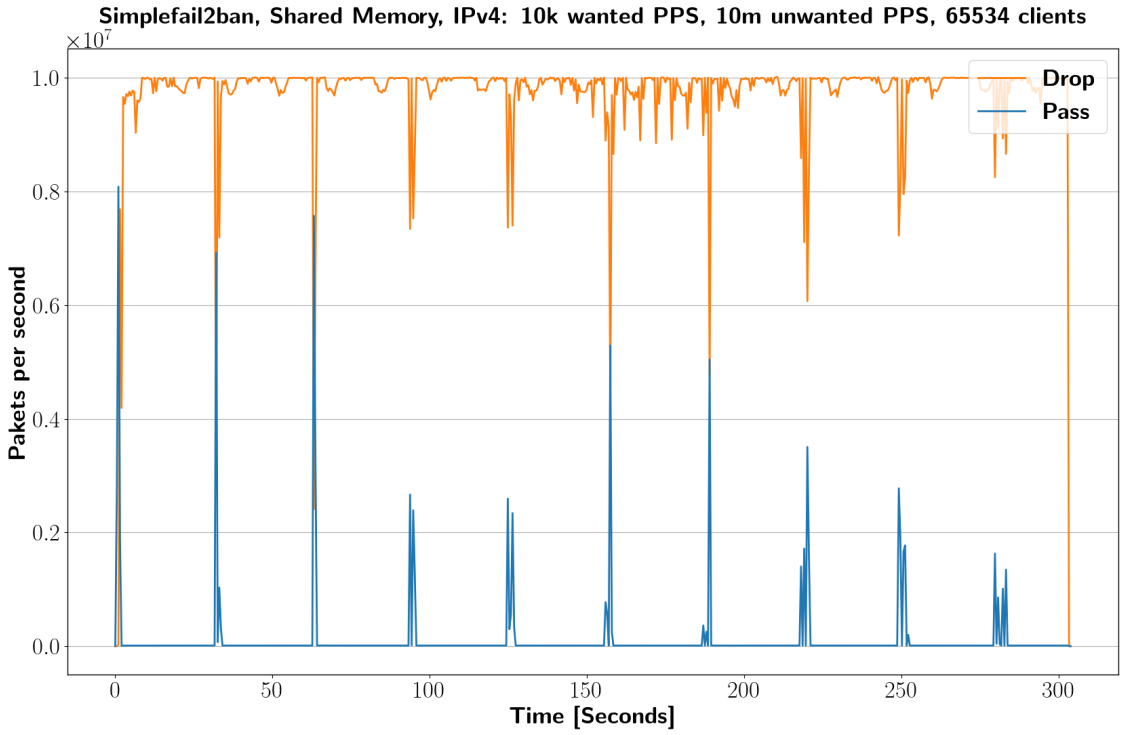
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
33	27.95	99.64	2.05	13.68

Figure 4.12: Some text



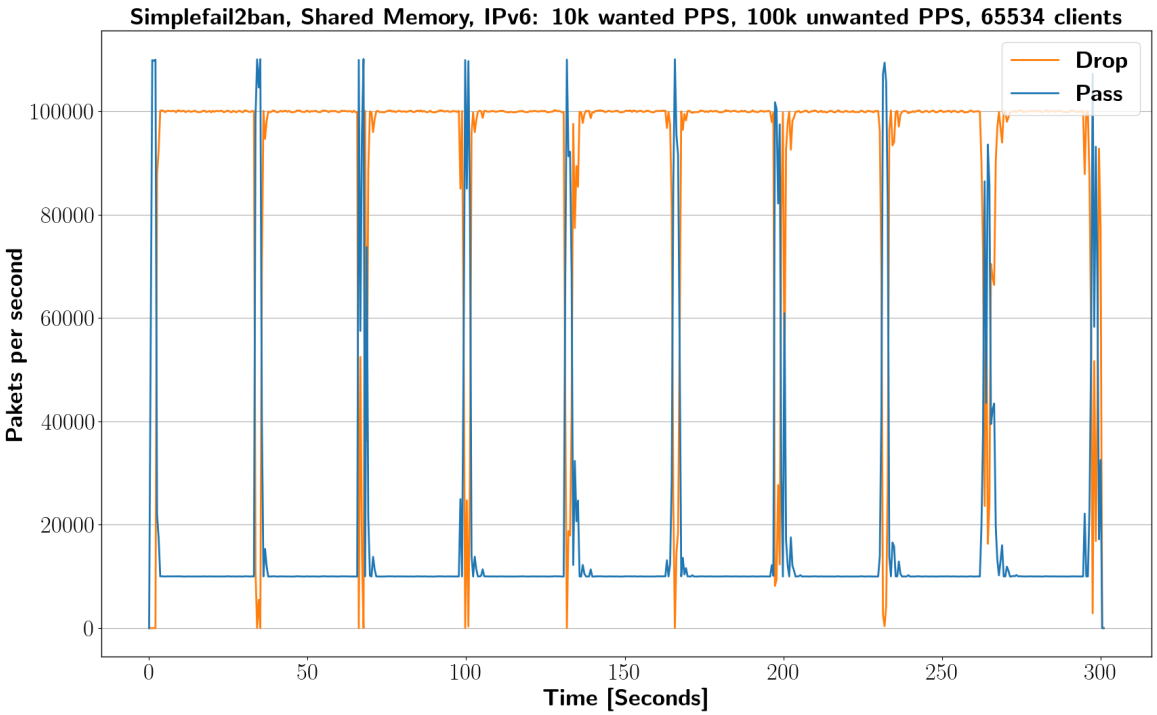
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
303	294.34	98.88	4.61	15.52

Figure 4.13: Some text



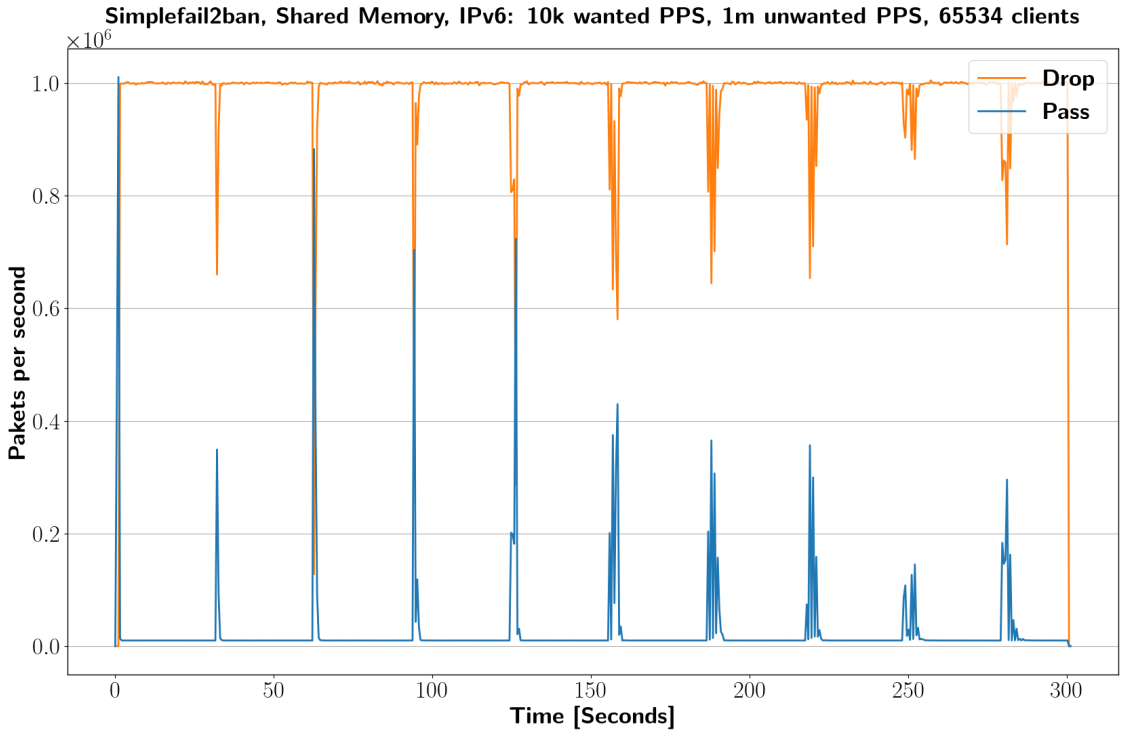
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2991.48	2948.51	96.72	7.69	25.59

Figure 4.14: Some text



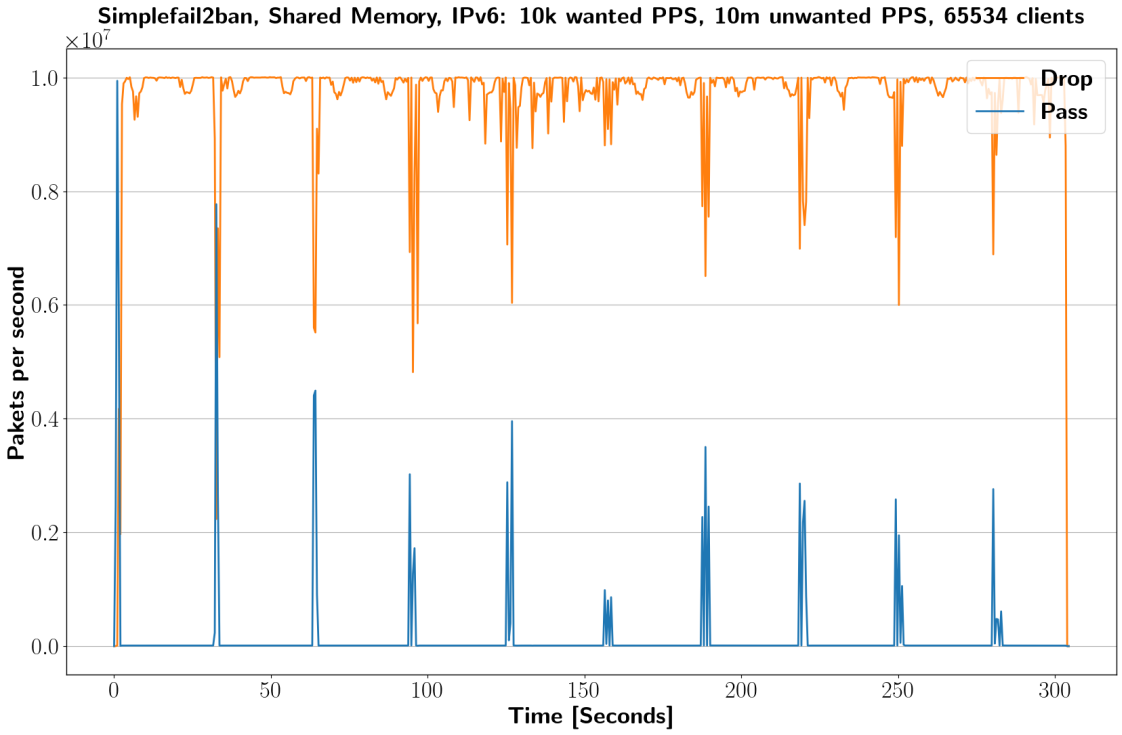
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
33	27.95	99.67	2.05	15.55

Figure 4.15: Some text



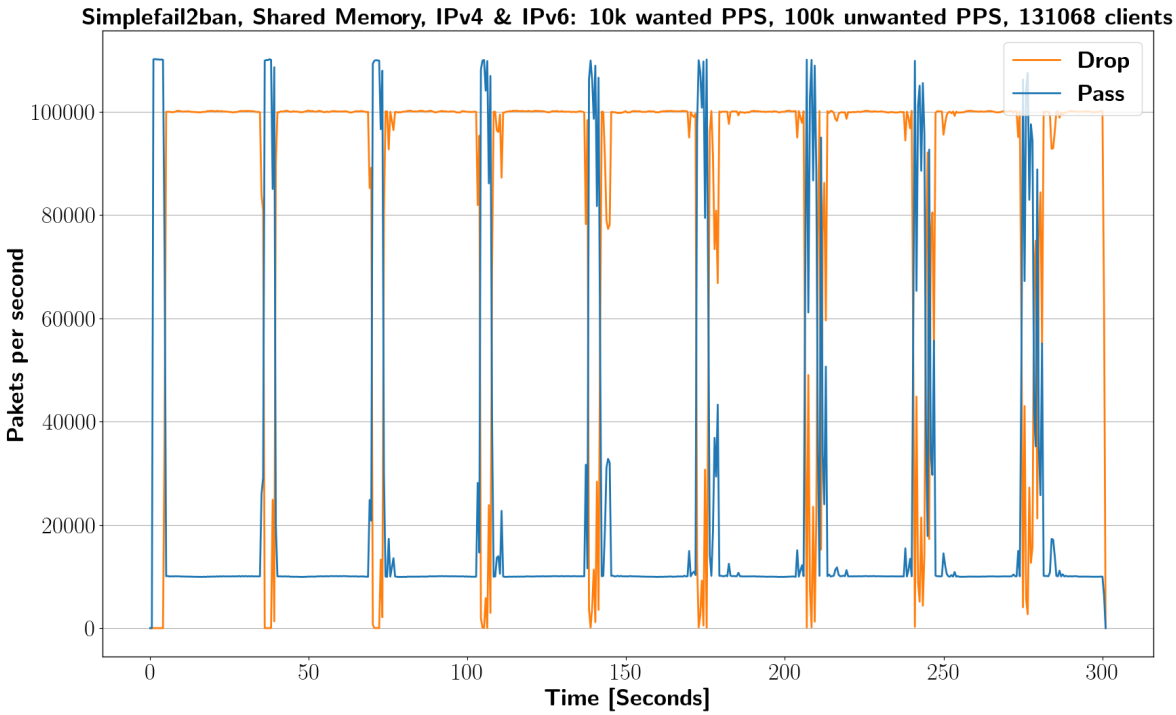
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
303	294.77	98.9	4.39	17.46

Figure 4.16: Some text



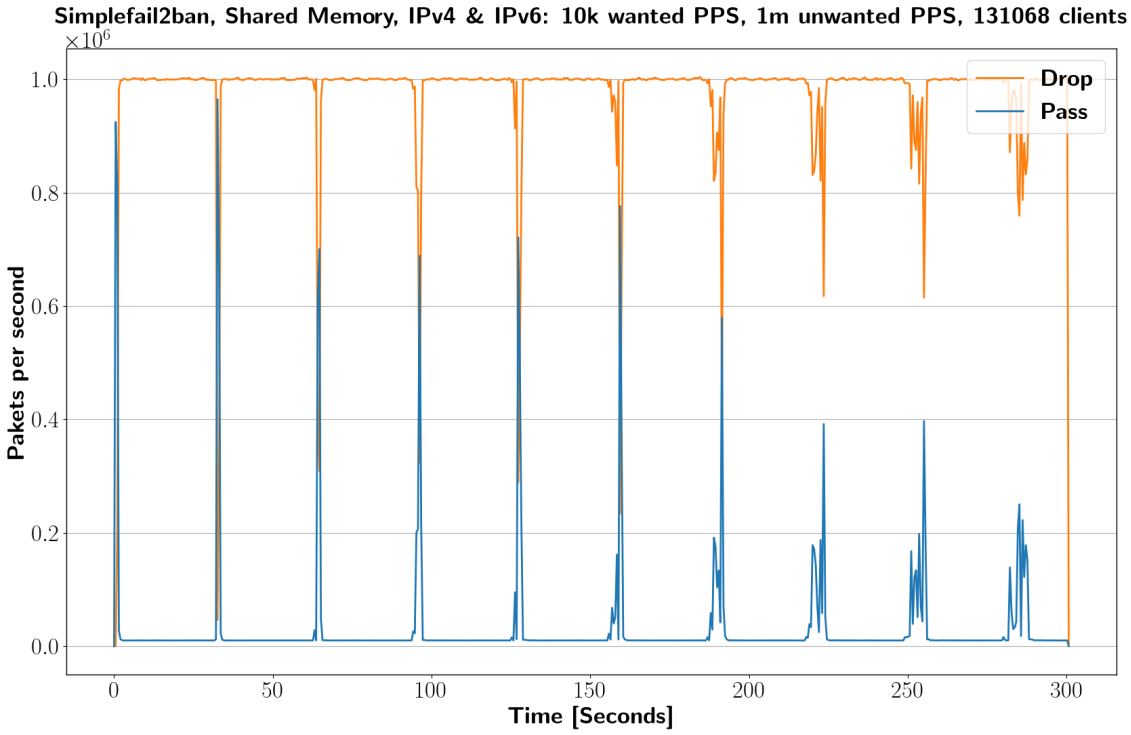
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2991.93	2947.1	98.66	8.88	32.36

Figure 4.17: Some text



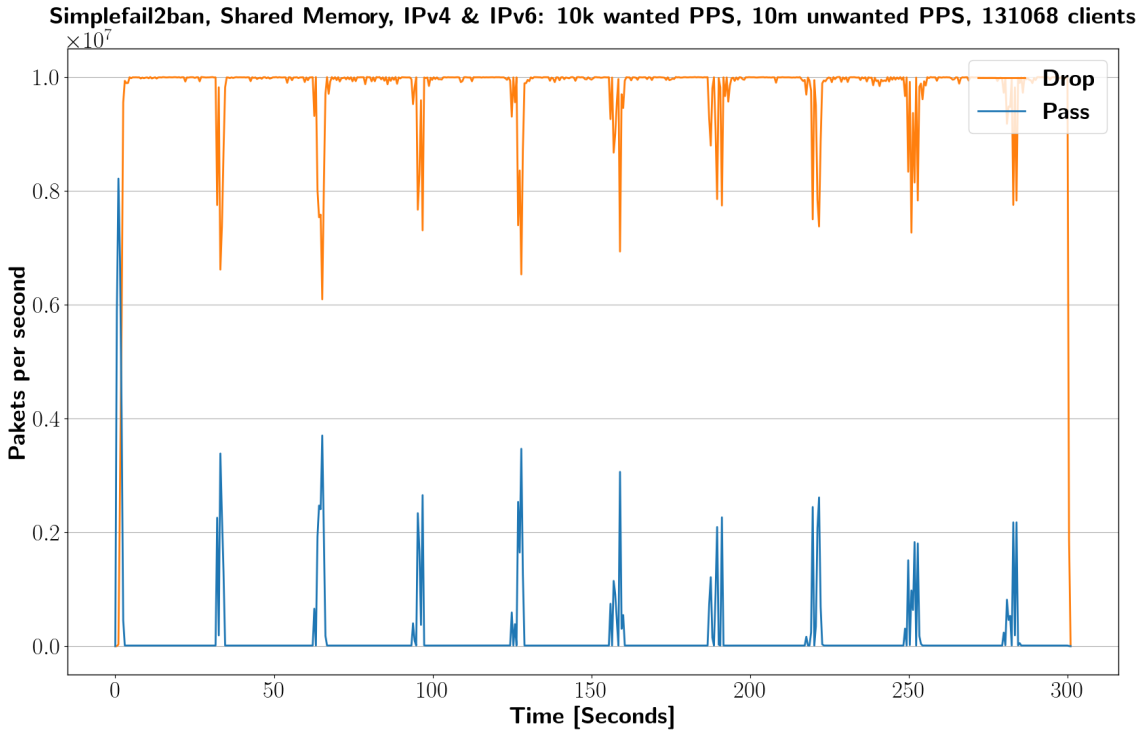
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
33	26.45	99.95	3.55	26.35

Figure 4.18: Some text



Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
303	293.13	99	5.95	27.66

Figure 4.19: Some text

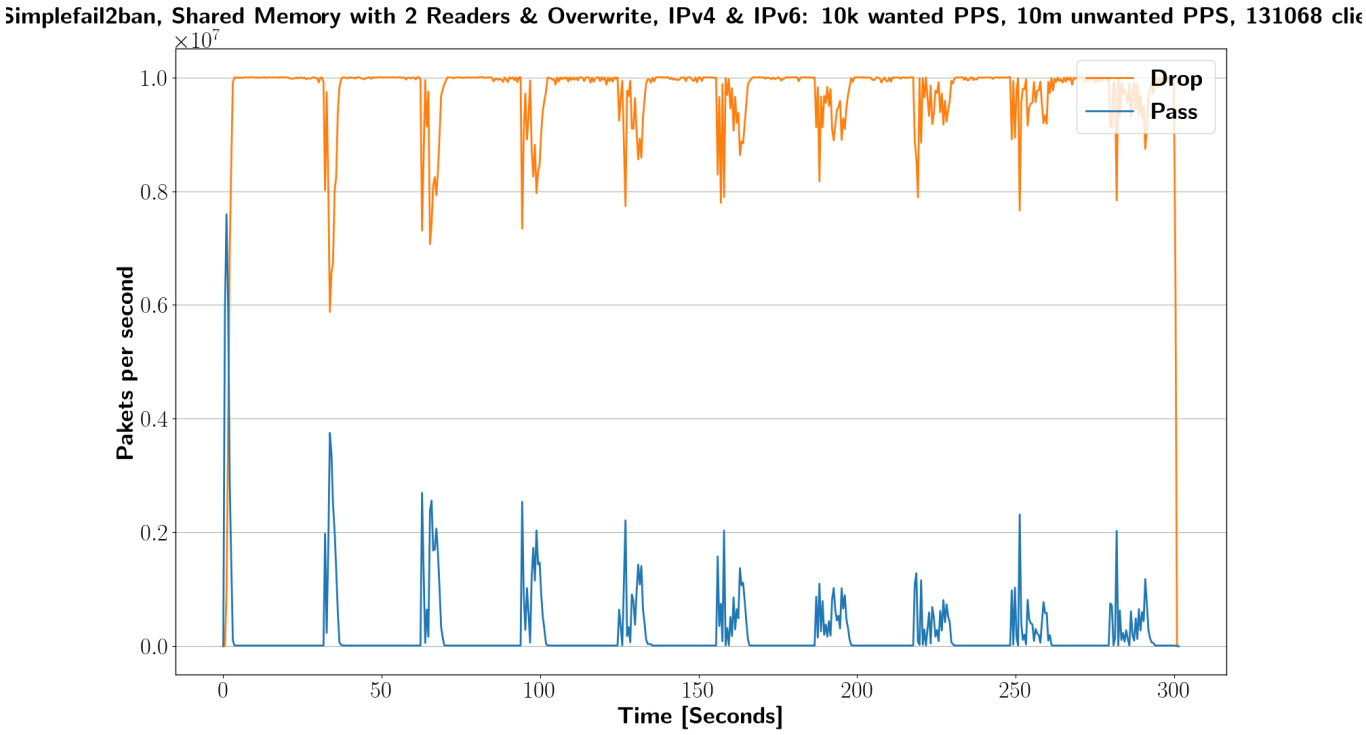


Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2992.44	2939.27	98.45	16.18	65.74

Figure 4.20: Some text

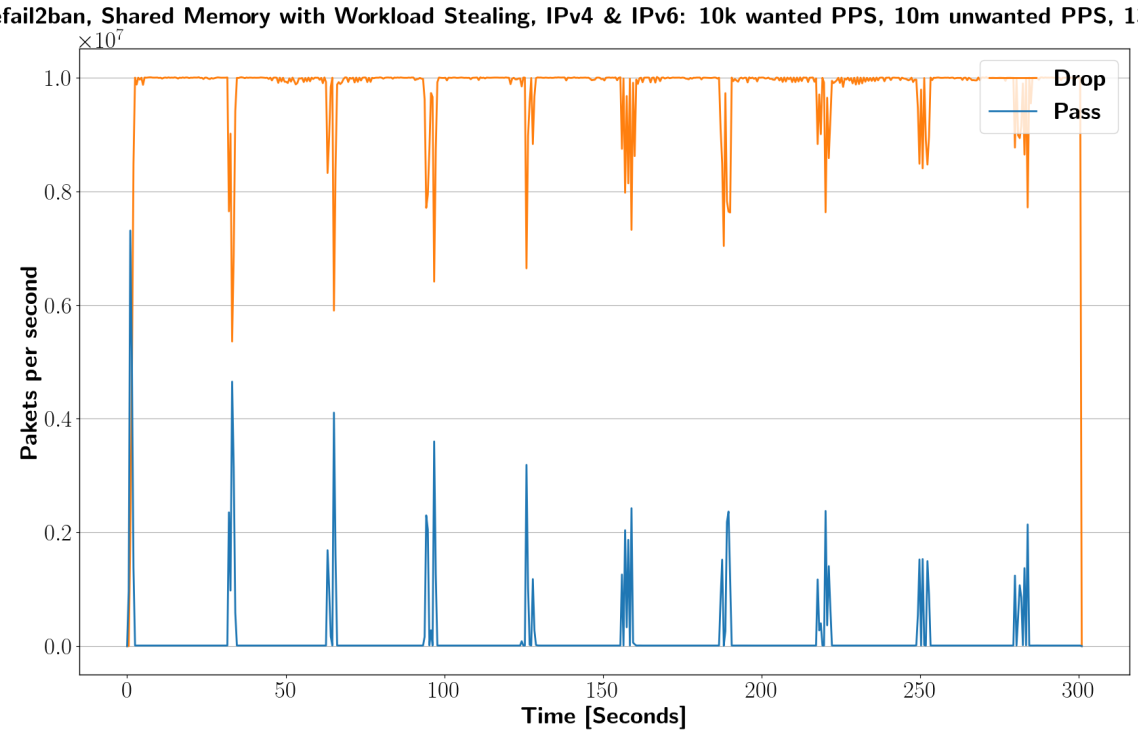
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2989.66	2906.36	97.44	17.67	77.19

Figure 4.21: Some text



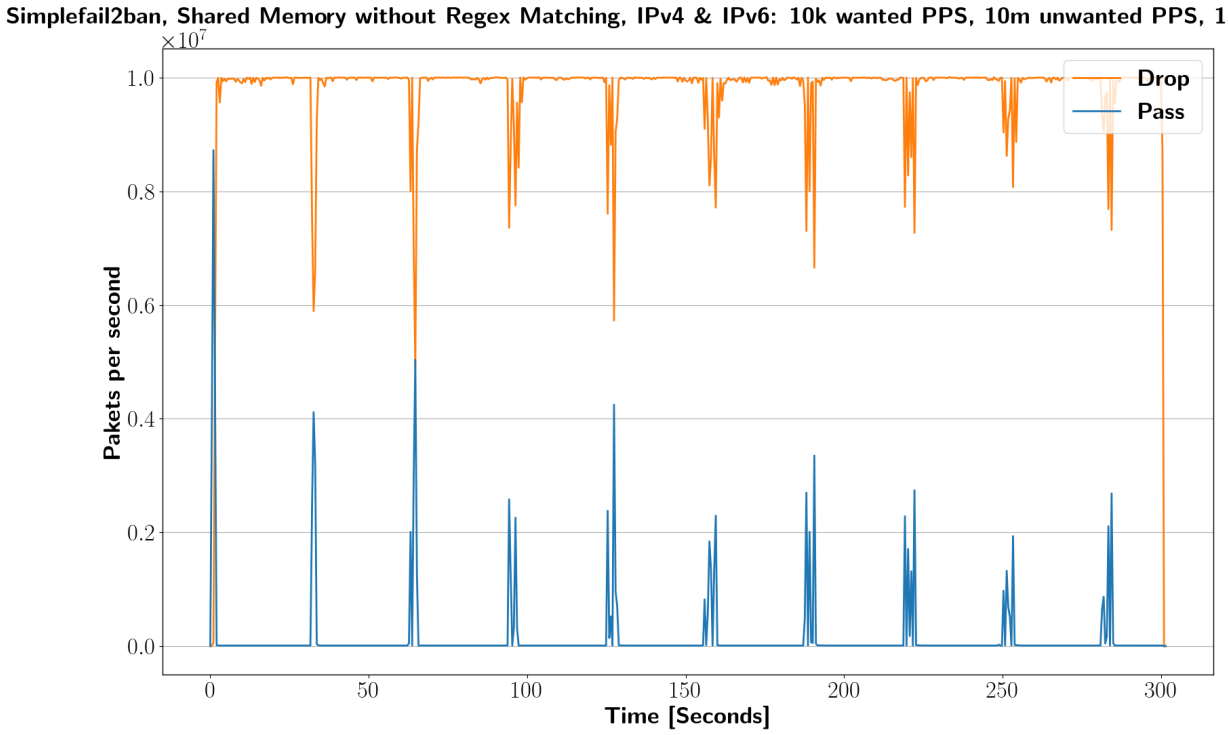
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2988.9	2914.39	97.73	17.66	79.13

Figure 4.22: Some text



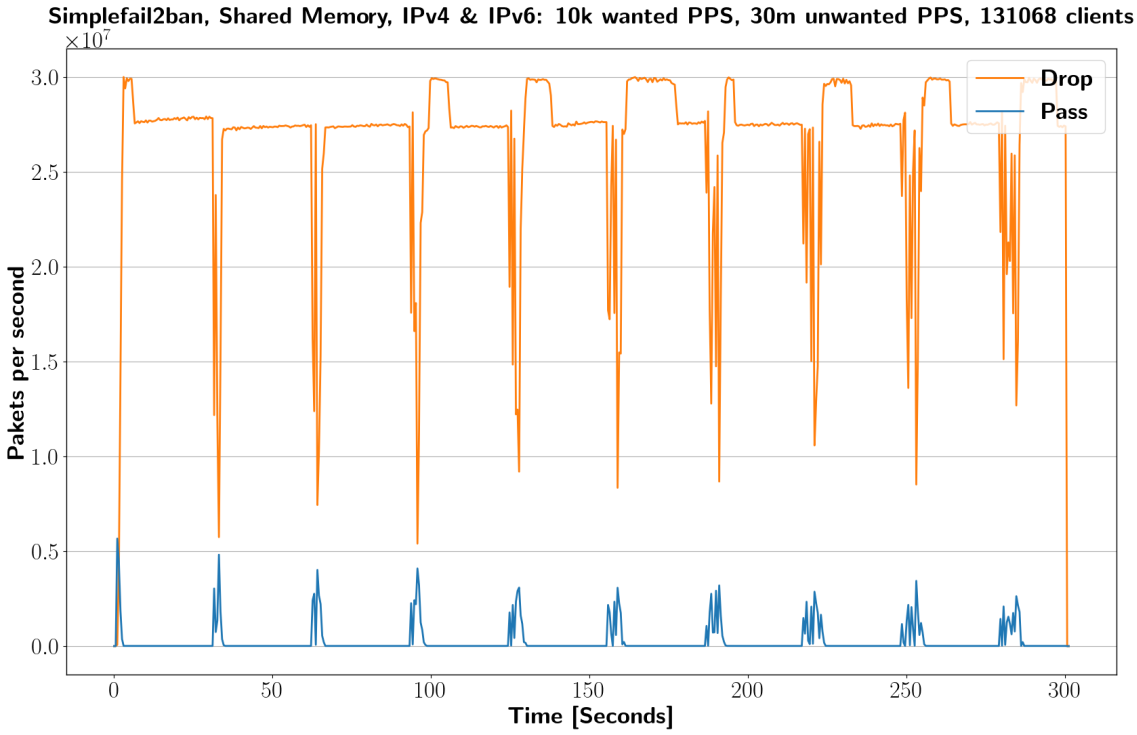
Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2991.23	2944.56	98.67	15.28	61

Figure 4.23: Some text



Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
2989.56	2941.9	98.63	13.29	34.04

Figure 4.24: Some text



Total packets [10^6]	Packets dropped [10^6]	Relative drop [%]	Log messages [10^6]	CPU [seconds]
8095.28	8017.72	99.13	21	76.23

Figure 4.25: Some text

5 Conclusion

List of Figures

List of Tables

List of Algorithms

A Abbreviations

HIDS	Host-based Intrusion Detection System
IPC	Inter-Process Communication
IPS	Intrusion Prevention System
OS	Operating System
PPS	Packets per Second

B Source Files

For the sake of the environment, no full source files are appended. However, all source files are available in a GitHub repository. The access to that repository is available from the second supervisor: Max Schrötter.

Bibliography

[1] Fail2Ban. Official Fail2Ban Website. `https://www.fail2ban.org`, 2011. Last visited on: 04.04.2023.

[2] James P Anderson. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company*, 1980.

[3] Dorothy E Denning. An intrusion-detection model. *IEEE Transactions on software engineering*, pages 222–232, 1987.

[4] Giovanni Vigna and Christopher Kruegel. Host-based intrusion detection. In *Handbook of Information Security*, pages 701–713. California State University, 2006.